Modelling Fusion calculus using HD-automata[⋆]

Gianluigi Ferrari[1], Ugo Montanari[1], Emilio Tuosto[1], Björn Victor[2], and Kidane Yemane[2]

[1] Dipartimento di Informatica, Università di Pisa, Italy.
[2] Dept. of Information Technology, Uppsala University, Sweden.

Abstract. We propose a coalgebraic model of the Fusion calculus based on HD-automata. The main advantage of the approach is that the partition refinement algorithm designed for HD-automata is easily adapted to handle Fusion calculus processes. Hence, the transition systems of Fusion calculus processes can be minimised according to the notion of observational semantics of the calculus. As a beneficial side effect, this also provides a bisimulation checker for Fusion calculus.

1 Introduction

Nominal calculi, process calculi with primitive mechanisms for local name generation, name exchange and scoping rules, have been successfully applied to specify and verify properties of *global computing systems*. Names provide a suitable abstraction to describe a variety of different computational phenomena such as mobility, localities, distributed object systems, security keys, session identifiers and son on. For instance, the π-calculus has been exploited for modelling and verifying a finite instance of the handover protocol of the Public Land Mobile Network [20]. Several properties of cryptographic protocols have been naturally expressed through spi-calculus specifications [1]. Nominal calculi also provide a basic programming model that has been incorporated in novel programming languages (see e.g. [6, 2]) and workflow languages for Web Service coordination [4, 16].

Verification via semantic equivalence provides a well established framework to reason about the behaviour of systems specified using nominal calculi. In this approach, checking behavioural properties is reduced to the problem of contrasting two system abstractions in order to determine whether their behaviours coincide with respect to a suitable notion of semantic equivalence. However, in the case of nominal calculi verification via semantic equivalence is intrinsically difficult. Indeed, when an unbound number of new names can be generated during execution, models of nominal calculi (e.g. labelled transition systems) tend to be infinite even in the simplest cases unless explicit mechanisms are introduced to deal with names.

Symbolic semantics [15, 3, 17] is a well established approach to finite state verification of nominal calculi. Symbolic semantics takes a *syntax-based* approach

[⋆] Work supported by the PROFUNDIS FET-GC project.

and generalises standard operational semantics by keeping track of equalities among names: transitions are derived in the context of such constraints. The main advantage of the symbolic semantics is that it yields a smaller transition system. The idea of symbolic semantics has been exploited to provide a convenient characterisation of *open bisimilarity* [25] and in the design of the corresponding bisimulation checker, the *Mobility WorkBench* (MWB) [27].

An alternative class of models for nominal calculi are the so-called *syntax-free* models where names are explicitly dealt with regardless of the syntactic structure of the calculi. *Indexed LTSs* [5] and *History Dependent automata* (HD-automata in brief) [18, 22, 19] are examples of syntax-free models of nominal calculi developed following the approach based on name permutations. HD-automata have an added value because they are a formal device for verification of nominal calcui modelled in a syntax-free context. This model encompasses the main features of nominal calculi, namely creation and deallocation of names, and accounts for a compact representation of process behaviour by collapsing states differing only for the renaming of local names. Basically, the "history" of the names appearing in the computation is explicitly represented so that it is possible to reconstruct the associations that have led to a given state. Clearly, if a state is reached in two different computations, different histories are assigned to its names [18, 22]. In [19], states of HD-automata have been equipped with name symmetries which further reduces the size of the automata and guarantee the existence of the minimal realization. The computation of the minimal automata is derived by exploiting a coalgebraic presentation of the partition refinement algorithm [7]. The minimisation algorithm for the early semantics of π-calculus has been implemented in the Mihda toolkit [10]. Since Mihda does not rely on a symbolic semantics, the number of states of the minimal HD-automaton is unnecessarily large due to the existence of different input transitions for different instantiations of the input variable. Hence, the integration of symbolic techniques and syntax-free models would provide more powerful verification methods. Notice that minimisation algorithms for syntax-based models have been already developed (e.g., for the open semantics of the π-calculus[23]).

In this paper we introduce a novel symbolic semantics for the Fusion calculus [21]. The Fusion calculus is a nominal calculus which extends the π-calculus with the capability of *fusing* names. When two names are fused then they can be used interchangeably. The Fusion calculus has been introduced as a simplification and generalization of the π-calculus. Apart from the theoretical interest Fusion calculus seem to arise naturally in the implementation of distributed systems, e.g. workflow languages for service coordination [24].

The main technical contribution of this paper is the development of a coalgebraic framework for the Fusion calculus equipped with the symbolic semantics. Our main result here is that the coalgebraic description of the minimisation algorithm for HD-automata smoothly extends to handle the Fusion calculus and behaves in accordance with the symbolic semantics: bisimilar processes are mapped together by the morphisms yielding the minimal HD-automaton.

The paper is structured as follows: In section 2 we describe HD-automata as a coalgebra over a category of *named-set* and the minimisation procedure for the HD-automata and in, Section 3, we provide a brief overview of Fusion calculus together with a new symbolic semantics. This will be followed by the HD-automata for Fusion calculus and how minimisation works for Fusion calculus HD-automata in Section 4, and we conclude in Section 5 with some conclusion, related work and some ideas for future work.

2 History Dependent Automata

Verification of concurrent and mobile systems specified using nominal calculi is intrinsically difficult since the state space can easily become extremely large or even infinite. *History Dependent automata* (HD-automata in brief) [22, 18, 19, 7] are an operational model for nominal calculi designed to address finite state verification. HD-automata can be seen as automata enriched by equipping states and transition with names. This permits to model name creation/deallocation or name extrusion which are typical linguistic mechanisms of nominal calculi.

A noteworthy fact is that names in states of HD-automata have *local meaning*, hence a compact representation of agent behaviour can be achieved by collapsing states that differ only for renaming of local names. Following [10], we provide a formal definition of HD-automata which basically differs from definitions of [10] in avoiding usage of dependent types. Indeed, the presentation in [10] aims at showing how the formal definition of the partition refinement algorithm for HD-automata guides and corresponds to its implementation Mihda [9]. Here, we build named sets on top of *permutation algebras*, namely we give a more abstract definition which focuses on the main mathematical ingredients necessary to describe the coalgebraic model and the related minimisation algorithm for Fusion calculus without taking into account implementation details.

Before giving the formal definitions, it is worth to collect some notations.

We consider a set $\mathcal{N}_\star$ made of a countable set of names $\mathcal{N}$ and a distinguished element $\star \notin \mathcal{N}$. We let $\mathrm{Aut}(\mathcal{N})$ be the set of bijective endofunctions on $\mathcal{N}$, i.e., the permutations of $\mathcal{N}$. Given a permutation ρ such that $\mathrm{dom}(\rho) \subset \mathcal{N}$ (where $\mathrm{dom}(\rho)$ is the domain of ρ), $\underline{\rho}$ is the automorphism on $\mathcal{N}_\star$ obtained by extending ρ on $\mathcal{N}_\star$ so that $\underline{\rho}(x) = x$ for any $x \in \mathcal{N}_\star \setminus \mathrm{dom}(\rho)$. The application of a function on names γ to an element e is written as $e\gamma$ and, when e is a set, it stands for the point-wise application of γ to the elements of e.

Definition 1 (Permutation algebras). *A* permutation algebra *is an algebra $\langle S, O\rangle$, where S is the carrier of the algebra and the set of operations O contains unary operators $\{\widehat{\rho} \mid \rho \in \mathrm{Aut}(\mathcal{N})\}$ such that the following axioms hold.*

$$\forall x \in S.\ x\widehat{id} = x, \qquad \forall x \in S \forall \rho_1, \rho_2 \in O.\ x\widehat{\rho_1;\rho_2} = (x\widehat{\rho_1})\widehat{\rho_2}.$$

Following [19], we can see the Fusion calculus as a permutation algebra: the carrier is the set of processes of Fusion calculus (up-to structural congruence) and the operations are name permutations interpreted as substitutions.

We introduce the notions of *named sets* and *named functions* which form the category **NS** of named sets, in terms of permutation algebras. Then we study the structure of **NS**.

Definition 2 (Named sets). *A* named set *(ns) is a pair* $\langle Q, \mathrm{g} \rangle$ *where;*

1. *Q is a permutation algebra;*
2. $\mathrm{g} : Q \to \bigcup_{N \in \wp_{fin}(\mathcal{N})} \mathrm{sym}(N)$ *assigns to any $q \in Q$ a group of permutations* $\mathrm{g}(q)$ *over a finite set of names such that $q = q\widehat{\rho}$, for any $\rho \in \mathrm{g}(q)$. The* names *of q, written as $|q|$, are defined as the domain of the permutations $\rho \in \mathrm{g}(q)$ while $\|q\|$ is the cardinality of $|q|$.*

We let D, E, F range over nss and, given $D = \langle Q, \mathrm{g} \rangle$, we write Q_D (resp. g_D) to denote Q (resp. g).

Definition 3 (Isomorphism of named sets). *Two nss D and E are* isomorphic *if there exists an* isomorphism between D and E, *namely a bijective function $s : Q_D \to Q_E$ such that, for any $d \in Q_D$ there is a bijective correspondence $n : |d| \to |s(d)|$ such that $\mathrm{g}_D(d) = \mathrm{g}_E(e); n$.*

Therefore, we consider nss up-to isomorphism so that two nss are considered equal whenever they have the same "structure" despite of having different underlying sets and different names associated to each element. It only matters the number of names and the symmetry associated to each element, namely, names are local to elements of nss.

Named functions basically are functions that preserve the structure of nss.

Definition 4 (Named functions). *Given two named sets D and E, a* named function *(nf) $H : D \to E$ is a pair $\langle h, \Sigma \rangle$ where $h : Q_D \to Q_E$ and $\Sigma : Q_D \to \wp_{fin}(Q_E \times \mathcal{N}_\star{}^{\mathcal{N}})$ are such that, for all $q \in Q_D$ and $(e, \sigma) \in \Sigma(q)$,*

1. *σ is injective, $\sigma(|e|) \subseteq |q| \cup \{\star\}$ and $\sigma(x) = x$, for any $x \in \mathcal{N} \setminus |e|$;*
2. *$\sigma; \mathrm{g}_D(q) \subseteq \Sigma_2(q)$, where $\Sigma_2(q) = \bigcup_{(e,\sigma) \in \Sigma(q)} \{\sigma\}$ and the permutations in $\mathrm{g}_D(q)$ are all meant to act as the identity on $\star$;*
3. *$\mathrm{g}_E(h(q)); \sigma = \Sigma_2(q)$.*

Named functions are ranged over by H, K and J. We write h_H and Σ_H for denoting the first and the second components of H. The intuition behind conditions 1, 2 and 3 naturally emerges when nfs are exploited to describe the transitions out of a certain state. Intuitively, elements in $\mathrm{h}_H(q)$ are the transitions out of q and $\Sigma_H(q)$ contains the mappings of names of target states of those transitions to names of q. Condition 1 ensures that any name in $|q|$ has a unique "meaning" along each transition in $\mathrm{h}_H(q)$ (injectivity of σ) and establishes that names in target states are either mapped on names of the source state or to $\star$, the distinguished name representing the generation of a new name along a transition. Condition 2 states that the group of the starting state q does not *generate* transitions which are not in $\Sigma_H(q)$. Finally, condition 3 states that any permutation is in the symmetry of $\mathrm{h}_H(q)$ iff, when applied to any σ mapping names of the transitions to those of q, yields a map in $\Sigma_H(q)$.

Definition 5 (Composition of named functions). *The* composition $H; K$ *of two nfs* $H : D \to E$ *and* $K : E \to F$ *is* $\langle \mathrm{h}_H; \mathrm{h}_K, \Sigma_H; \Sigma_K \rangle$ *where* $\Sigma_H; \Sigma_K : Q_D \to \wp_{\mathit{fin}}(Q_F \times \mathcal{N}_\star{}^{\mathcal{N}})$ *is such that*

$$\Sigma_H; \Sigma_K : q \mapsto \bigcup_{(e,\sigma) \in \Sigma_H(q)} \{(f, \sigma'; \sigma') \mid (f, \sigma') \in \Sigma_K(h(q))\}.$$

Proposition 1. *In Definition 5,* $H; K$ *is a nf. Composition of nfs is associative and has identities.*

Proof. The proof proceeds as the corresponding proof in [11]. □

Definition 6 (Category of named sets). *The* category **NS** *has nss as objects and nfs as morphisms.*

The basic charecteristics of **NS** are collected in Proposition 2.

Proposition 2 (Structure of NS). *The category* **NS** *has initial object, terminal object, and finite powerset functor defined as follows:*

1. *the initial object given by* $\bot = \langle \emptyset, \emptyset \rangle$*;*
2. *the terminal object is given by* $I = \langle \{*\}, * \mapsto \emptyset \rangle$*;*
3. *the powerset functor on* ***NS*** *is obtained by lifting the covariant powerset functor* $\wp_{\mathit{fin}}(\_)$ *on* **Set***, namely* $\wp_{\mathit{fin}}(D) = \langle \wp_{\mathit{fin}}(D_Q), \mathrm{g} \rangle$*, where, given* $Q \subseteq Q_D$*,* $\mathrm{g}(Q) = \{\rho \mid \rho \text{ is a permutation over } \bigcup_{q \in Q} |q|\} \wedge Q\rho = Q$.

Definition 7 (Pairing of named sets). *Given two nss* D *and* E*, the* pairing $D \otimes E$ of D and E *is defined as* $D \otimes E = \langle Q_D \times Q_E, \mathrm{g} \rangle$ *where*

– $\mathrm{g} : Q_D \times Q_E \to \bigcup_{N,M \in \wp_{\mathit{fin}}(\mathcal{N})} (\mathrm{sym}(N) + \mathrm{sym}(M))$ *such that* $\mathrm{g}(d, e) = \{\rho_1 + \rho_2 \mid \rho_1 \in \mathrm{g}_D(d) \wedge \rho_2 \in \mathrm{g}_E(e)\}$.

Formally, $D \otimes E$ is not a ns because the range of $\mathrm{g}_{D \otimes E}$ is not the union of symmetries over a finite set of names. However, observing that for any $(d, e) \in Q_D \times Q_E$, $\mathrm{g}(d, e)$ is a symmetry on $|d| + |e|$, we can find a ns whose group function maps (d, e) to a symmetry over as many names as in $|d| + |e|$ and whose permutations correspond bijectively to the permutations in $\mathrm{g}(d, e)$. Notice that, since nss are considered up-to isomorphism, it does not matter which set of names is chosen for any pair (d, e).

Definition 8 (HD-automata). *Fixed a ns of labels* L*, a* HD-automaton *over* L *is a coalgebra for* $T_L(D) = \wp_{\mathit{fin}}(L \otimes D)$.

We emphasise that nfs provide the formal mean to describe a generic step of the iterative minimisation algorithm. Intuitively, nfs map states of the automaton in equivalence classes containing those states considered equivalent.

Definition 9 (Kernel of named functions). *The* kernel of a nf $H : E \to F$ *(written as* $\ker H$*) is the ns* D *such that:*

1. $Q_D = \ker \mathrm{h}_H$ considered as permutation algebra where for all $A \in Q_D$ and $\rho \in \mathrm{Aut}(\mathcal{N})$, $A\rho$ is the element-wise application of ρ to A;
2. the group of $A \in \ker \mathrm{h}_H$ is $\mathrm{g}_F(\mathrm{h}_H(a))$, for $a \in A$.

In [7, 10] the normalisation functor for the early semantics of π-calculus has been introduces. In this context, the concept of *redundancy* relies on the concept of *active names* because of the presence of freshly generated names. We generalise this concept by means of *redundant transitions.* Generally, redundant transitions are transitions describing behaviours that can be matched by other transitions in the bisimulation game. Redundant transitions occur when HD-automata are built out of a nominal calculus. During this phase, it is not possible to decide which are the redundant transitions[1]. Therefore, all the transitions are taken when HD-automata are built and redundant ones are removed during the minimisation. This is achieved by means of the most important operation of **NS** namely *normalisation* which basically gets rid of reduntandant transitions.

Definition 10 (Normalisation functor). *A* normalisation functor N *is any functor such that $N(D)$ is isomorphic to a subset of D.*

The minimisation algorithm on a T_L coalgebra $(D, K : D \to T_L(D))$ is specified by the equations 1 and 2 below.

$$H_{(0)} \overset{\mathrm{def}}{=} \langle q \mapsto \bot, q \mapsto \emptyset \rangle, \quad \text{where } \mathrm{dom}(H_{(0)}) = D \tag{1}$$

$$H_{(i+1)} \overset{\mathrm{def}}{=} K; N(T(H_{(i)})), \tag{2}$$

where N is a normalisation functor and $T : \mathbf{NS} \to \mathbf{NS}$ is the functor defined as

$$T(D) = \begin{cases} T_L(D) & D \in \mathrm{obj}(\mathbf{NS}) \\ \langle \mathrm{h}, \Sigma \rangle & D = \langle h_D, \Sigma_D \rangle \in \mathbf{NS}(E, F) \text{ for } E, F \in \mathrm{obj}(\mathbf{NS}) \end{cases}$$

where, given $B \in \wp_{\mathrm{fin}}(L \otimes E)$,

$$\begin{aligned} \mathrm{h}(B) &= \{\langle l, \mathrm{h}_D(q) \rangle \mid \langle l, q \rangle \in B\} \\ \Sigma(B) &= \{\langle l, \mathrm{h}_D(q), \sigma; \sigma' \rangle \mid \langle l, q, \sigma' \rangle \in B \land \langle l, q', \sigma' \rangle \in \Sigma_D(q)\}. \end{aligned}$$

All the states of automaton K are initially considered equivalent, indeed, $\ker H_0$ gives rise to a single equivalence class containing the whole $\mathrm{dom}(K)$. At the generic $(i+1)$-th iteration, as specified in (2), the image through $H_{(i)}$ of the i-th iteration is composed with K as stated by the definition of functor T, then the normalisation functor removes the redundant transitions. The algorithm builds the minimal realisation $\bar{H}$ of (finite) HD-automata by constructing (an approximation of) the final coalgebra morphism. The kernel of $\bar{H}$ yields the equivalence classes where equivalent states are grouped in the same class.

The proof of the convergence of the algorithm is based on the observation that T is a monotonic functor over a finite chains. In order to establish this, we must give a way of saying when two nfs are the same.

[1] In general, to decide redundancy is as difficult as deciding bisimilarity.

Definition 11 (Equivalence of named functions). *Two nfs $H : D \to E$ and $K : D' \to E'$ are* equivalent *when* $\ker H$ *is isomorphic to* $\ker K$ *via the bijections n and s (see Definition 3) and for all $q \in D$,* $\mathrm{g}_E(\mathrm{h}_H(q)); n = \mathrm{g}_{E'}(\mathrm{h}_K(s(q)))$.

Definition 12 (Order of named functions). *Let $H : D \to E$ and $K : D \to F$ be two nfs, H* is less than of equal to *K (written as $H \preceq K$) if, and only if,*

- $Q_{\ker H}$ *is coarser than* $Q_{\ker K}$ *and* $\forall A \in Q_{\ker H}.\forall B \in Q_{\ker K}.\ B \subseteq A \Rightarrow \mathrm{g}_{\ker H}(A) \subseteq \mathrm{g}_{\ker K}(B)$.
- $\forall A \in Q_{\ker H}.\forall B \in Q_{\ker K}.\ \forall q \in A \cap B. \Sigma_H(q) \subseteq \Sigma_K(q)$.

Proposition 3. *Relation $\preceq$ is a pre-order and $H \preceq K \wedge K \preceq H$ implies* H and K equivalent.

Proof. The first condition of Definition 12 and $H \preceq K \wedge K \preceq H$ imply that $\ker H$ is isomorphic to $\ker K$. It remains to prove that the hypothesis implies the last condition of Definition 11.

Assume that there is $q \in \mathrm{dom}(H)$ such that $\mathrm{g}_{\mathrm{cod}(H)}(\mathrm{h}_H(q)) \neq \mathrm{g}_{\mathrm{cod}(K)}(\mathrm{h}_K(q))$. Then, for all $\sigma \in \Sigma_H(q)$,

$$\mathrm{g}_{\mathrm{cod}(H)}(\mathrm{h}_H(q)); \sigma \neq \mathrm{g}_{\mathrm{cod}(K)}(\mathrm{h}_K(q)); \sigma. \tag{3}$$

By Definition 12, $\Sigma_H(q) = \Sigma_K(q)$ since $H \preceq K \wedge K \preceq H$. Moreover, conditions on nfs (Definition 4) imply that $\mathrm{g}_{\mathrm{cod}(H)}(\mathrm{h}_H(q)); \sigma = \Sigma_{\mathrm{h}_H}(q)$ and $\mathrm{g}_{\mathrm{cod}(K)}(\mathrm{h}_K(q)); \sigma = \Sigma_{\mathrm{h}_K}(q)$ that contradicts (3). □

Monotonicity is preserved by composition of named functions:

Lemma 1. *Let H and K be two nfs such that $H \preceq K$. For any ns J, if* $\mathrm{cod}(J) = \mathrm{dom}(H) = \mathrm{dom}(K)$ *then $J; H \preceq J; K$.*

Finally, we can prove the convergence of the iterative algorithm:

Theorem 1 (Convergence). *The iterative algorithm described by (1) and (2) is convergent on finite state HD-automata whenever the normalisation functor N is monotone on nfs.*

Proof. By construction, $\wp_{\mathrm{fin}}(\_)$ is monotone, hence T is monotone because it is the composition of two monotone functors, moreover, by monotony of T and Lemma 1, maps $\mathcal{M} : H \mapsto K; T(H)$ is monotone and finite. Finally, all nfs chains having finite domain are finite, hence, the iterative algorithm defined in (1) and (2) converges to the maximal fix-point of $\mathcal{M}$. □

The proofs of Theorem 1 mimics that in [10]. The only difference is that the theorem in [10] is proved only for the case of the early semantics of π-calculus, while here, the result is extended to the general case of finite HD-automata, with the only additional assumption that the normalisation functor is monotone.

$$\textsc{pref}\ \frac{-}{\alpha\,.\,P \xrightarrow{\alpha} P} \qquad \textsc{sum}\ \frac{P \xrightarrow{\gamma} P'}{P+Q \xrightarrow{\gamma} P'} \qquad \frac{P \xrightarrow{\gamma} P'}{P \mid Q \xrightarrow{\gamma} P' \mid Q}\ \textsc{par}$$

$$\textsc{com}\ \frac{P \xrightarrow{u\tilde{x}} P',\ \ Q \xrightarrow{\overline{u}\tilde{y}} Q',\ \ |\tilde{x}| = |\tilde{y}|}{P \mid Q \xrightarrow{\{\tilde{x}=\tilde{y}\}} P' \mid Q'} \qquad \frac{P \xrightarrow{\varphi} P',\ z\,\varphi\,x,\ z \neq x}{(z)P \xrightarrow{\varphi\setminus z} P'\{x/z\}}\ \textsc{scope}$$

$$\textsc{pass}\ \frac{P \xrightarrow{\gamma} P',\ z \notin \mathrm{n}(\gamma)}{(z)P \xrightarrow{\gamma} (z)P'} \qquad \frac{P \xrightarrow{(\tilde{y})a\,\tilde{x}} P',\ z \in \tilde{x} - \tilde{y},\ a \notin \{z, \overline{z}\}}{(z)P \xrightarrow{(z\tilde{y})a\,\tilde{x}} P'}\ \textsc{open}$$

$$\textsc{match}\ \frac{P \xrightarrow{\gamma} P'}{[x=x]P \xrightarrow{\gamma} P'} \qquad \frac{P \equiv P' \quad P \xrightarrow{\gamma} Q \quad Q \equiv Q'}{P' \xrightarrow{\gamma} Q'}\ \textsc{struct}$$

Table 1. Transition rules for the Fusion calculus

3 Syntax and Semantics of the Fusion calculus

We briefly recollect the syntax and operational semantics of the Fusion calculus. For lack of space, we refer the reader to [21] for further details. In Section 3.1 we present a new canonical symbolic semantics.

Definition 13 (Fusion calculus syntax). *The* free actions *ranged over by α,* fusion action *ranged over by φ,* actions *ranged over by γ, and the* agents *ranged over by $P, Q, \ldots$, are defined by*

$$\begin{array}{ll|l} \alpha ::= u\tilde{x} & \overline{u}\tilde{x} \\ \gamma ::= \alpha & \varphi \end{array} \qquad \begin{array}{lllll} P ::= \mathbf{0} & \mid\ \gamma\,.\,Q & \mid\ Q + R & \mid\ Q \mid R \\ & \mid\ (x)Q & \mid\ [x=y]Q & \mid\ A\langle\tilde{x}\rangle, \end{array}$$

where x, y, u, $v\ldots$ range over $\mathcal{N}$ and represent communication channels, which are also the values transmitted. An input action $u\tilde{x}$ means "input objects along the port u and replace $\tilde{x}$ with these objects". Note that input does not entail binding. The output action $\overline{u}\tilde{x}$ means "output the objects $\tilde{x}$ along the port u". A fusion action $\{\tilde{x} = \tilde{y}\}$ represents an obligation to make $\tilde{x}$ and $\tilde{y}$ equal everywhere, limited by the scope of the names involved.

Definition 14 (Fusion calculus semantics). *The labelled transition system of Fusion calculus is the least relation satisfying the inference rules in Table 1.*

Use of the SCOPE rule entails a substitution of the scoped name z for a nondeterministically chosen name x related to it by φ. For the purpose of the equivalence defined below it will not matter which such x replaces z. The only rule dealing with bound actions is OPEN. Using structural congruence, pulling the relevant scope to top level, we can still infer e.g. $P \mid (x)ayx\,.\,Q \xrightarrow{(x)ayx} P \mid Q$ using PREF and OPEN (provided $x \notin \mathrm{fn}(P)$, otherwise an alpha-conversion is necessary).

Definition 15 (Hyperbisimulation [21]). *A* fusion bisimulation *is a binary symmetric relation $\mathcal{S}$ between agents such that $(P, Q) \in \mathcal{S}$ implies:*

If $P \xrightarrow{\gamma} P'$ *with* $\mathrm{bn}(\gamma) \cap \mathrm{fn}(Q) = \emptyset$*, then* $Q \xrightarrow{\gamma} Q'$ *and* $(P'\sigma_\gamma, Q'\sigma_\gamma) \in \mathcal{S}$*. Agents* P *and* Q *are* fusion bisimilar*, written* $P \stackrel{\centerdot}{\sim} Q$*, if* $(P, Q) \in \mathcal{S}$ *for some fusion bisimulation* $\mathcal{S}$*. A* hyperbisimulation *is a substitution closed fusion bisimulation.*

Theorem 2. [21] *Hyperequivalence is the largest congruence in fusion bisimilarity.*

3.1 Canonical Symbolic Semantics of Fusion calculus

Having briefely presented Fusion calculus syntax together with its concrete semantics, we provide a new symbolic semantics of Fusion calculus which lend itself to coalgebraic modeling through HD-automata. The *canonical* symbolic semantics for the Fusion calculus is defined along the lines of symbolic semantics for the π-calculus [25, 23]. Symbolic semantics are often used to give efficient characterizations of bisimulation equivalences for value-passing calculi.

In Table 2 we present the symbolic transition system where structurally equivalent agents are considered the same. Like in [25] a symbolic transition is of the form $P \xmapsto{M,\gamma} Q$, where M is the enabling condition of the action γ in the sense that M represents the equalities a minimal substitution σ_M must make true in order for $P\sigma_M$ to perform the corresponding action in the original labelled transition system. σ_M is the *substitutive effect* of M: an idempotent substitution s.t. $\sigma_M(x) = \sigma_M(y)$ iff $M \Rightarrow x = y$. We generalise substitutive effects to actions, where σ_α is the identity substitution.

In Table 2 we write MN for denoting the concatenation of M and N. Following Pistore and Sangiorgi's work [23], our transition rules apply substitutions to the continuation of a transition: like [23], a substitution σ_M, making the condition for the transition true, and in addition a substitution σ_γ, the substitutive effect of the action, is applied to the right-hand side of the transition. The motivation for this is to make the definition of bisimulation simpler and more in line with the algorithms used in the HD framework (see Section 4). We show later in this section that bisimulation using the symbolic semantics coincides with the original non-symbolic version.

Using canonical substitutions gives us pleasant properties like the following:

Lemma 2. *If* $P \xmapsto{M,\gamma} P'$*, then* $\gamma = \gamma\sigma_M$ *and* $P' = P'\sigma_M = P'\sigma_\gamma = P'\sigma_M\sigma_\gamma$*.*

The definition of symbolic hyperbisimulation is similar to that of symbolic open bisimulation [25, 23], but does not have the complication of distinctions.

Definition 16 (Symbolic hyperbisimulation). *A binary symmetric process relation* $\mathcal{S}$ *is a* symbolic hyperbisimulation *if* $(P, Q) \in \mathcal{S}$ *implies:*
If $P \xmapsto{M,\gamma} P'$ *with* $\mathrm{bn}(\gamma) \cap \mathrm{fn}(Q) = \emptyset$ *then* $Q \xmapsto{N,\gamma'} Q'$ *such that*
  $M \Rightarrow N$*,* $\gamma = \gamma'\sigma_M$*,* *(note* $\gamma = \gamma\sigma_M$*)*
  and $(P', Q'\sigma_M) \in \mathcal{S}$ *(note* $P' = P'\sigma_M$*).*
P *is* symbolically hyperequivalent *to* Q*, written* $P \simeq Q$*, if* $(P, Q) \in \mathcal{S}$ *for some symbolic hyperbisimulation* $\mathcal{S}$*.*

$$\text{PREF}\ \frac{-}{\alpha\,.\,P \xmapsto{\emptyset,\alpha} P\sigma_\alpha} \qquad \text{SUM}\ \frac{P \xmapsto{M,\gamma} P'}{P+Q \xmapsto{M,\gamma} P'} \qquad \frac{P \xmapsto{M,\gamma} P'}{P \mid Q \xmapsto{M,\gamma} P' \mid Q\sigma_M\sigma_\gamma}\ \text{PAR}$$

$$\text{SCOPE}\ \frac{P \xmapsto{M,\varphi} P',\ z\,\varphi\,x,\ z \neq x,\ z \notin \mathrm{n}(M)}{(z)P \xmapsto{M,\varphi\setminus z} P'\{x/z\}} \qquad \frac{P \xmapsto{M,\gamma} P' \qquad M' = M[x=y]}{[x=y]P \xmapsto{M',\gamma\sigma_{[x=y]}} P'\sigma_{M'}}\ \text{MATCH}$$

$$\text{PASS}\ \frac{P \xmapsto{M,\gamma} P',\ z \notin \mathrm{n}(M,\gamma)}{(z)P \xmapsto{M,\gamma} (z)P'}$$

$$\frac{P \xmapsto{M,(\tilde{y})a\,\tilde{x}} P',\ z \in \tilde{x} - \tilde{y},\ a \notin \{z,\overline{z}\},\ z \notin \mathrm{n}(M)}{(z)P \xmapsto{M,(z\tilde{y})a\,\tilde{x}} P'}\ \text{OPEN}$$

$$\text{COM}\ \frac{P \xmapsto{M,u\tilde{x}} P',\ \ Q \xmapsto{N,\overline{v}\tilde{y}} Q',\ \ |\tilde{x}| = |\tilde{y}|,\ \ L = MN[u=v],\ \varphi = \{\tilde{x}=\tilde{y}\}\sigma_L}{P \mid Q \xmapsto{L,\varphi} (P' \mid Q')\sigma_L\sigma_\varphi}$$

Table 2. Canonical symbolic transition system for the Fusion calculus

Since the symbolic semantics applies the substitutive effects, we can leave most of that out of the bisimulation definition. It is still necessary to apply substitution corresponding to the stronger condition, σ_M, to the label and continuation of the transition of Q. (Note that $Q'\sigma_M = Q'\sigma_M\sigma_\gamma$.)

Theorem 3. *$P \sim Q$ iff $P \simeq Q$*

Proof. See appendix.

4 From Fusion calculus to HD-automata

This section describes how agents of Fusion calculus can be mapped onto HD-automata and what normalisation means for the HD-automata for Fusion calculus. We first introduce labels and transitions and then define the normalisation functor for Fusion calculus. We note the monotonicity of the this functor guaranty the convergence of the minimisation algorithm on finite HD-automata that correspond to Fusion calculus agents. We conclude the section with an informal discussion on the correspondence between hyperequivalence and minimisation. In order to keep the coalgebraic presentation as simple as possible, we limit to a monadic version of Fusion calculus where tuples in communication actions carry a single name.

Though not increasing the expressiveness of the calculus, polyadicity would obscure the main picture of the coalgebraic presentation with cumbersome technical details. (A mapping of the polyadic Fusion calculus to HD-automata is given in [26].)

The labels of the canonical symbolic semantics of Fusion calculus are consists of enabling conditions and actions; both of them can be represented as nss.

Let M be the ns $\langle \{\bullet\}, \mathrm{g} \rangle$ where $\mathrm{g} = \{id_{x,y}, exch_{x,y}\}$, namely, g contains the identity and the exchanging permutation on the two names in M i.e, $|\bullet| = \{x, y\}$.

Definition 17 (Matching *named-set*). *A* matching named-set *is a ns of the form* $\mathtt{M} = M \underbrace{\otimes \ldots \otimes}_{n \geq 0} M$*, also written as* M^n *(recall, that pairing treats names in a component as distinguished from those in other components;* $\mathtt{M}_i$ *is the i-th component of* $\mathtt{M}$*).*

Given a name substitution $\sigma \in {\mathcal{N}_\star}^{\mathcal{N}}$*, the* interpretation of $\mathtt{M}$ in σ *is* $[\![\mathtt{M}]\!]_\sigma$ *and is defined as*

$$\sigma(x_1) = \sigma(y_1) \wedge \cdots \wedge \sigma(x_n) = \sigma(y_n),$$

where, for any $i = 1, \ldots, n$*,* $\{x_i, y_i\}$ *are the names in* $\mathtt{M}_i$*.*

As notation, M^0 is the ns $\langle \{\bullet\}, \bullet \mapsto \emptyset \rangle$ namely, the singleton ns where $|\bullet| = \emptyset$. Basically, enabling conditions are represented by tupling matching nss each representing a fusion of two names. The interpretation of $\mathtt{M}$ under σ is the statement constraining the names x_i and y_i of any component $\mathtt{M}_i$ to be identified once they are interpreted through σ. Notice that the interpretation of M^0 under any substitution always hold and, indeed, it represent the trivial condition $[x = x]$. (Any substitution σ such that $[\![\mathtt{M}]\!]_\sigma$ holds true is said to be compatible with $\mathtt{M}$.)

Definition 18 (Labels for Fusion calculus). *Let* $\mathtt{M}$ *be a matching ns and Lab be the set* $\{\mathtt{tau}, \mathtt{in}, \mathtt{out}, \mathtt{fuse}\}$*. The* nss of labels for Fusion calculus *are the nss* $\mathtt{M} \otimes \langle Lab, \mathrm{g} \rangle$ *where* $\mathrm{g} : Lab \to \mathcal{N}_\star$ *is such that*

$\|\mathtt{tau}\| = 0$
$\|\mathtt{fuse}\| = 2 \wedge \mathrm{g}(\mathtt{fuse})$ *has the identity and the exchanging permutations*
$\|\mathtt{in}\|, \|\mathtt{out}\| \leq 2 \wedge \mathrm{g}(\mathtt{in}),\ \mathrm{g}(\mathtt{out})$ *has only the identity permutation.*

$\mathtt{M}$ *is called the* enabling part *and* g *is called the* action part *of labels for Fusion calculus.*

Hereafter, L stands for the ns of labels for Fusion calculus and $K : D \to T_L(D)$ is an HD-automaton. (Roughly, L represents labels of transitions of the canonical symbolic semantics of Fusion calculus.)

Let P be a Fusion calculus agent, $K[P]$ denotes the coalgebraic specification of the HD-automaton associated to P, namely a T_L-coalgebra such that $\mathrm{dom}(K[P]) = D[P]$ and $\mathrm{cod}(K[P]) = T_{NS}(D[P])$.

Let $Q_{D[P]}$ denote the set of Fusion calculus processes reacheable from P, namely

$$Q_{D[P]} \overset{\mathrm{def}}{=} \{P\} \ \cup \bigcup_{P \xmapsto{M,\gamma} P'} \{P'\} \cup Q_{D[P']}.$$

It is trivial to equip $Q_{D[P]}$ with a *named-set* structure, indeed for any $q \in Q_{D[P]}$, the group component $\mathrm{g}_{D[P]}(q)$ is the identity on $\mathrm{fn}(q)$.

Function $\mathrm{h}_{K[P]}$ associates, to each state, its outgoing transitions and is defined as

$$\mathrm{h}_{K[P]}(q) = \{\langle l, q', \sigma\rangle \mid q \xmapsto{M,\gamma} q' \wedge l \text{ corresponds to } M, \gamma\}$$

where, for any $\langle l, q', \sigma\rangle \in \mathrm{h}_{K[P]}(q)$ σ maps the names in $\mathrm{fn}(q')$ that correspond to those in $\mathrm{fn}(q)$ and names generated in the transition to $\star$ (and similarly for the names of l). Recall, indeed, that the names in q' and l are *local*, hence, even though they are syntactically equal in the Fusion calculus transition system, they must be considered different in HD-automata states.

The HD-automaton obtained by this definition is a T_L-coalgebra by construction. Observe that infinite HD-automatacan be obtained using the construction above, however, there are interesting classes of Fusion calculus agents that generate finite HD-automata: this is the case of *finitary* agents. The *degree of parallelism* $\deg(P)$ of a Fusion calculus agent P is defined as follows:

$$\begin{array}{rr} \deg(\mathbf{0}) = 0 & \deg(\alpha\,.\,P) = 1 \\ \deg((x)P) = \deg(P) & \deg(P \mid Q) = \deg(P) + \deg(Q) \\ \deg([x=y]P) = \deg(P) & \deg(P + Q) = \max(\deg(P), \deg(Q)) \end{array}$$
$$\deg(A\langle y_1, \ldots, y_n\rangle) = \deg(P[^{y_1,\ldots,y_n}/_{x_1,\ldots,x_n}]), \quad \text{if } A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} P$$

Agent P is *finitary* if $\max\{\deg(P') \mid P \xmapsto{M_1,\gamma_1} \cdots \xmapsto{M_i,\gamma_i} P'\} < \infty$. In [19, 22] the following result has been proved:

Theorem 4 (Theorem 47 of [19]). *Let P be a finitary agent. Then the HD-automaton $K[P]$ is finite.*

Let us now define the normalisation functor for Fusion calculus.

Definition 19 (Redundancy of labels). *Let $\langle l, q, \sigma\rangle$ and $\langle l', q, \sigma'\rangle$ be two hdt of K. Assuming that the matching ns of l (resp. l') is* `M` *(resp.* `M'`*), l is* redundant wrt l' *iff l and l' have the same action part and $[\![M]\!]_\sigma$ logically implies $[\![M']\!]_{\sigma'}$ but not vice versa.*

Definition 20 (Redundant transitions). *Let $q \in Q_D$ be a state of K, an hdt $\langle l_1, q_1, \sigma_1\rangle$ is* redundant (abbreviated as rhdt) for q *if there is $\langle l_2, q_1, \sigma_2\rangle$ in $\Sigma_K(q)$ such that l_1 is redundant wrt l_2 and, for a substitution σ accomplishing with the interpretation of the enabling part of l_1, $\sigma_2; \sigma = \sigma_1$.*

The intuition is that $t = \langle l_1, q_1, \sigma_1\rangle$ is dominated by another transition $t' = \langle l_2, q_1, \sigma_2\rangle$ reaching the same target state as t and with the same label but having

- enabling conditions weaker than those of t and,
- under the conditions of t, the names associated to the label of t' are the same as those of t.

Definition 21 (Normalisation functor for Fusion calculus). *The* normalisation functor for Fusion calculus *denoted by* $\mathtt{N} : \boldsymbol{NS} \to \boldsymbol{NS}$*, is defined as follows:*

$$\mathtt{N}(D) = \begin{cases} \langle \mathrm{h}, \Sigma\rangle & D = \langle h_D, \Sigma_D\rangle \in \boldsymbol{NS}(\wp_{\mathit{fin}}(L \otimes E), \wp_{\mathit{fin}}(L \otimes F)) \;\; \textit{for } E, F \in \mathit{obj}(\boldsymbol{NS}) \\ D & \textit{otherwise.} \end{cases}$$

where, for $B \in \wp_{fin}(L \otimes E)$,

$$\begin{aligned} \mathrm{h}(B) &= \{\langle l, q\rangle \mid \;\; \nexists \langle l, q, \sigma\rangle \textit{ rhdt in } \Sigma(B)\} \\ \Sigma(B) &= \{\langle l, q, \sigma\rangle \in B \mid \langle l, q, \sigma\rangle \textit{ not rhdt in } \} \end{aligned}$$

Basically, N filters those transitions out of a given state q that are redundant because of the presence of another transition having weaker conditions on names.

Proposition 4. *The functor* $\mathtt{N}$ *is monotonic on nfs.*

Theorem 5. *The minimisation algorithm described in Section 2 converges on finite HD-automaton for Fusion calculus.*

Proof. By the monotonicity of T and $\mathtt{N}$ and Theorem 1.

Let us remark that normalisation trough $\mathtt{N}_H$ is based on Definition 16 of symbolic hyperbisimulation. Indeed, redundancy conditions for Fusion calculus simply are the conditions in Definition 16 relating the enabling part and the action part of bisimilar transitions. Hence, a tight relationship can be established between Fusion calculus hyperbisimulation and the outcome of the minimisation algorithm.

Theorem 6 (Minimisation and hyperbisimulation). *Two Fusion calculus processes are hyperbisimilar iff they have the same minimal realisation*

Proof. (Sketch.) On the one hand, given two bisimilar Fusion calculus agents P and Q, if corresponding HD-automata $K[P]$ and $K[Q]$ are finite, their minimal realisations, say $\bar{K}[P]$ and $\bar{K}[Q]$, achieved by the minimisation algorithm are equivalent. Namely, $\ker \bar{K}[P] = \ker \bar{K}[Q]$ which implies that their corresponding classes have the same symmetries. On the other hand, if $K[P]$ and $K[Q]$ are finite and they both have the same minimal realisation, say $\bar{K}$, then P and Q are bisimilar. Basically, this is due to the fact that, the transitions out of a state in $K[P]$ (resp. $K[Q]$) have a corresponding transition in the behaviour of P (resp. Q). By construction of $\bar{K}$ all the possible transitions of P have a matching non rhdt in $\bar{H}$, hence can be matched by Q as well and vice versa. □

5 Conclusions

Related work. This work is related to the work of Ferrari et al [8–10], and Cattani and Sewell [5] where they both follow syntax-independent model approach to the operational semantics of process calculi. The former goes further to introduce a minimization procedure of transition systems for nominal calculi in a coalgebraic setting but only treated an early semantics of π-calculus. Another related work worth noting is the work of Fiore and Staton [12], and Fabio et al [13] where they provide a formal comparison of several operational semantics of nominal calculi.

In this paper we take the work of Ferrari et al [8–10] approach further to give the same functionality for the Fusion calculus. Hyperbisimulation in Fusion calculus is more sophisticated than early and late bisimulation of π-calculus which was studied in the above mentioned work because of the closure under all substitution required by hyperbisimulation. We solve these technical challenges by providing a new symbolic semantics of Fusion calculus and conservatively extending HD-automata. The presentation of the minimisation algorithm given in Section 2 differs from those [7, 10] because we neatly distinguish between two phases. The first phase "immerges" the initial HD-automaton to the current iteration of the algorithm, while the second phase applies a suitable normalisation functor for removing redundant transitions.

In the future we wish to take this research further and to deal with open semantics of π-calculus. Open semantics of π-calculus is complicated because extruded names need to be recorded and kept distinct from all other names under renaming. On the theoretical side, it would be interesting to study how this work is related to a presheaf model of open semantics of π-calculus as in [14] and other approaches as in [12, 13].

References

1. M. Abadi and A. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. *Information and Computation*, 148(1):1–70, January 1999.
2. N. Benton, L. Cardelli, and C. Fournet. Modern Concurrency Abstractions for C#. *ACM Transactions on Programming Languages and Systems*, 26(5):269–304, Sept. 2004.
3. M. Boreale and R. De Nicola. A Symbolic Semantics for the π-calculus. *Information and Computation*, 126(1):34–52, April 1996.
4. R. Bruni, H. Melgratti, and U. Montanari. Theoretical Foundations for Compensations in Flow Composition Languages. In *Annual Symposium on Principles of Programming Languages POPL*, 2005. To appear.
5. G. L. Cattani and P. Sewell. Models for name-passing processes: Interleaving and causal (extended abstract). In *Proceedings of the Fifteenth Annual IEEE Symposium on Logic in Computer Science, LICS 2000*, pages 322–333. IEEE Computer Society Press, 2000.
6. S. Conchon and F. Le Fessant. Jocaml: Mobile Agents for Objective-Caml. In *International Symposium on Agent Systems and Applications*, pages 22–29, Palm Springs, California, Oct. 1999.
7. G. Ferrari, U. Montanari, and M. Pistore. Minimizing Transition Systems for Name Passing Calculi: A Co-algebraic Formulation. In M. Nielsen and U. Engberg, editors, *Foundations of Software Science and Computation Structures*, volume 2303 of *Lecture Notes in Computer Science*, pages 129–143. Springer-Verlag, 2002.
8. G. Ferrari, U. Montanari, and M. Pistore. Minimizing transition systems for name passing calculi: A co-algebraic formulation. In M. Nielsen and U. H. Engberg, editors, *Proceedings of FoSSaCS 2002*, volume 2303 of *LNCS*, pages 129–143. Springer, Apr. 2002.
9. G. Ferrari, U. Montanari, and E. Tuosto. From Co-algebraic Specifications to Implementation: The Mihda toolkit. In F. de Boer, M. Bonsangue, S. Graf, and W. de Roever, editors, *Symposium on Formal Methods for Components and Objects*, volume 2852 of *Lecture Notes in Computer Science*, pages 319 – 338. Springer-Verlag, November 2002.

10. G. Ferrari, U. Montanari, and E. Tuosto. Coalgebraic Minimisation of HD-automata for the π-Calculus in a Polymorphic λ-Calculus. *Theoretical Computer Science*, 2004. To appear.
11. G. Ferrari, U. Montanari, and E. Tuosto. Modular Verification of Systems via Service Coordination. In *Monterey Workshop 2004*, October 2004. To appear on the workshop post-proceedings.
12. M. Fiore and S. Staton. Comparing operational models of name-passing process calculi. In J. Adamek, editor, *Proc. CMCS'04*, ENTCS. Elsevier, 2004.
13. F. Gadducci, M. Miculan, and U. Montanari. About permutation algebras and sheaves (and named sets, too!). Technical Report UDMI/26/2003/RR, Department of Mathematics and Computer Science, University of Udine, 2003.
14. N. Ghani, B. Victor, and K. Yemane. Relationally staged computation in the π-calculus. In *Procedings of CMCS 2004*, number 106, 11 in ENTCS, pages 105–120, 2004.
15. M. Hennessy and H. Lin. Symbolic Bisimulations. *Theoretical Computer Science*, 138(2):353–389, February 1995.
16. C. Laneve and G. Zavattaro. Foundations of Web Transactions. In *Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science, 2005. To appear.
17. H. Lin. Complete Inference Systems for Weak Bisimulation Equivalences in the π-Calculus. *Information and Computation*, 180(1):1–29, January 2003.
18. U. Montanari and M. Pistore. History Dependent Automata. Technical report, Computer Science Department, Università di Pisa, 1998. TR-11-98.
19. U. Montanari and M. Pistore. π-Calculus, Structured Coalgebras, and Minimal HD-Automata. In M. Nielsen and B. Roman, editors, *Mathematical Foundations of Computer Science*, volume 1983 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000. An extended version will be published on Theoretical Computer Science.
20. F. Orava and J. Parrow. An algebraic verification of a mobile network. *Formal Aspects of Computing*, 4(5):497–543, 1992.
21. J. Parrow and B. Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *Proceedings of LICS '98*, pages 176–185. IEEE, Computer Society Press, July 1998.
22. M. Pistore. *History Dependent Automata*. PhD thesis, Computer Science Department, Università di Pisa, 1999.
23. M. Pistore and D. Sangiorgi. A Partition Refinement Algorithm for the π-Calculus. *Information and Computation*, 164(2):467–509, 2001.
24. U. Roxburgh. Biztalk orchestration: Transactions, exceptions, and debugging, 2001. Microsoft Corporation. Available at http://msdn.microsoft.com/library/en-us/dnbiz/html/bizorchestr.asp.
25. D. Sangiorgi. A Theory of Bisimulation for the π-Calculus. *Acta Informatica*, 33(1):69–97, 1996.
26. E. Tuosto, B. Victor, and K. Yemane. Polyadic History-Dependent Automata for the Fusion Calculus. Technical Report 2003-62, Department of Information Technology, Uppsala, Sweden, December 2003. Available at `http://www.it.uu.se/research/reports/`.
27. B. Victor and F. Moller. The Mobility Workbench — A Tool for the π-Calculus. In D. Dill, editor, *Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*, pages 428–440. Springer-Verlag, 1994.

A Proof sketches for Section 3

Lemma 3. *For any Fusion calculus agent P, if $P \xrightarrow{\gamma} P'$, then $P\sigma \xrightarrow{\gamma\sigma} P'\sigma$, for any substitution σ.*

In the remainder of this section we establish the correspondence between symbolic hyperequivalence (Definition 16) and the standard hyperequivalence (Definition 15) by proving Theorem 3: $P \sim Q$ iff $P \simeq Q$.

Lemma 4.

1. *$\sigma\sigma_{R\sigma} = \sigma_R\rho$, for any substitution σ and some ρ, where R is an equivalence relation.*
2. *If $M \Rightarrow N$ then for any substitution σ, $M\sigma = N\sigma\rho$, for some substitution ρ.*
3. *$\sigma_R\sigma_{S\sigma_R} = \sigma_S\sigma_R$, where R and S are equivalence relations.*

Lemma 5.

1. *If $P \xmapsto{M,\gamma} P'$, then $P\sigma \xmapsto{M\sigma,\gamma\sigma} P'\sigma\sigma_{M\sigma}\sigma_{\gamma\sigma}$.*
2. *if $P\sigma \xmapsto{N,\gamma'} P'$, then $P \xmapsto{M,\gamma} P''$ with $M\sigma \Leftrightarrow N$, $\gamma\sigma = \gamma'$, and $P' = P''\sigma\sigma_N\sigma_{\gamma'}$.*

Proof. By transition induction, using Lemma 4. □

Lemma 6. *$P \simeq Q$ implies $P\sigma \simeq Q\sigma$, for any substitution σ.*

Proof. Straightforward diagram chasing, using Lemmas 4 and 5. □

Lemma 7.

1. *If $P \xmapsto{M,\gamma} P'$, then $P\sigma_M \xrightarrow{\gamma} P''$ s.t. $P' = P''\sigma_\gamma$;*
2. *if $M \Rightarrow N$ and $P\sigma_M \xrightarrow{\gamma} P'$, then $P \xmapsto{N,\gamma'} P''$ such that $\gamma = \gamma'\sigma_M$ and $P'\sigma_\gamma = P''\sigma_M$.*

Proof. Again by transition induction, using Lemmas 4 and 3. □

Proof of Theorem 3:

⇒: by showing $\mathcal{S} = \{(P,Q) : P \sim Q\}$ is a symbolic hyperbisimulation, using Lemmas 7 and 4. □

⇐: We already have closure under substitution (Lemma 6), and show that $\mathcal{S} = \{(P,Q) : P \simeq Q\}$ is a fusion bisimulation using Lemmas 7 and 4. □